应用于射电天文的高效实时管道数据流传输与处理技术

张萌[1,2]，张海龙[1,2,3,4*]，王杰[1,2,4]，李健[1]，冶鑫晨[1,4]，王万琼[1]，李嘉[1]，王博群[1,2]，张亚州[1,2]

（1. 中国科学院新疆天文台，新疆 乌鲁木齐，830011；2. 中国科学院大学，北京，100049；
3. 中国科学院射电天文重点实验室，江苏 南京 210008；4.国家天文科学数据中心，北京，100101）

摘 要：针对超宽带及多波束接收系统海量天文信号实时高效传输与处理问题，对基于 FPGA+GPU 的主流终端设备软件系统进行了测试分析，超宽带接收设备要求终端系统软件能够在更宽带宽，更高时间、频率分辨率条件下，实现实时数据流传输与处理。结合大口径射电观测设备未来发展方向提出了利用高速并行环形缓冲区实现数据流缓存、基于 GPU 集群实现数据流实时处理、基于 BeeGFS 实现分布式并行数据存储，模块化构建射电天文信号传输管道软件的设计思路。

关键词： 数据传输与处理；射电天文；实时； FPGA+GPU

中图分类号：P161.4　**文献标识码：**A　**文章编号：**

0 引言

多波束和 PAF 接收技术的发展使得天文观测设备获取数据的能力不断提升，射电天文观测数据正在以一种实时、海量、持续的模式增长，终端系统中海量天文信号的高效传输、实时处理，需要异构的分布式计算与存储平台作为支撑，数据传输过程中首先将接收到的信号进行数字化，实现通道划分、RFI 消减等预处理，这些处理可在 ROACH[1]、SNAP[1]、RFSoC[2]等系列开发板卡上执行。硬件板卡与服务器平台通过高速网络连接，将数据打包并通过网络传输至 GPU 服务器进行复杂算法处理。基于 FPGA+GPU 的混合架构平台对软件性能提出了更高要求，高速实时传输天文数据包时需要考虑高 IO 环境下的网络数据包丢失问题，Linux 内核在处理宽带网络流量时有很大的开销，由于网络协议是 UDP，系统必须处理丢失、无序和重复的数据包，否则会造成相位信息缺失，针对脉冲星数据，丢包可对脉冲星的周期预测和折叠等产生严重影响。实现 CPU 与 GPU 之间数据高速流转时，GPU 高效实时数据

---

*基金项目：国家自然科学基金(11873082,11803080,12003062)；中国科学院青年创新促进会；国家重点研发计划(2018YFA0404704)；天文财政专项；国家天文科学数据中心，中科院科学数据中心体系资助.

收稿日期：xxxx-xx-xx；修订日期：xxxx-xx-xx

作者简介：张萌,女,博士研究生.研究方向：数据密集型研究. Email：zhangmeng@xao.ac.cn

通讯作者：张海龙,正高级工程师.研究方向：数据密集型研究.Email:zhanghailong@xao.ac.cn

[1] https://casper.ssl.berkeley.edu/wiki/SNAP

处理阶段,CPU 与 GPU 之间通信速率可能会造成的性能瓶颈,也是需要考虑并解决的问题。

射电天文数字终端系统中可实现数据包高速率实时传输与预处理的程序称为管道[3]。管道是一种编程实现，数据被链式的连接在一起，全部同时在其内部流动。管道包含多个处理单元，每个单元执行特定任务，最基础和主要的功能是提供缓存模块，暂时存储数据，实现数据的实时流转。应用于射电天文异构终端系统的管道数据传输主要包括三个阶段，数据接收阶段、基于 CPU/GPU 的处理阶段、数据输出阶段，通过缓冲模块实现从数据接收到写入磁盘流程中的高效传输，通过调用不同阶段的程序模块实现数据流传输过程中的实时处理。

文章研究并分析了可部署在射电望远镜终端系统上的数据管道，提出了基于 FPGA+GPU 的动态、高性能实现海量射电天文数据流实时传输、处理和监控的管道设计思路，为未来新疆 110 米射电望远镜（QTT）[4]开发定制多功能实时数据流传输与处理管道软件提供参考。

1 现有射电天文数据流管道软件

针对射电天文数据的实时高速传输,国外已取得了系列研究成果,提供了开源 CPU/GPU 数据流管道软件，包括 GUPPI_daq[2]、PSRDADA[3]、Hashpipe、Bifrost、Kotekan[4]、Pelican[5]、Cobalt，国内相关研究现正处于起步阶段。

GUPPI_daq 是 GUPPI 数据采集子系统，通俗灵活、使用简单，已有十多年的历史；PSRDADA 和 Hashpipe 均为高吞吐量流数据处理而设计,最初分别应用于 Parkes[5]和 GBT[6]；Bifrost 是用于快速管道开发的开源软件框架，专门为射电天文实时数据流处理而设计，用于 LWA[7]后端；Kotekan 作为 CHIME[8]高度优化的流数据处理框架，更注重效率和吞吐量；Pelican 提供了高度抽象的应用程序接口，为静态、准实时处理目的而设计；Cobalt 管理通过网络和 GPU 的数据流，主要应用于 LOFAR 波束合成。

1.1 GUPPI_daq

应用于 GBT 的 GUPPI_daq，是脉冲星数字终端 GUPPI 的数据采集软件，GUPPI_daq 实现了数据流实时接收和处理,并以 PSRFITS 格式记录文件,实时处理选项包括基础模式、

[2] https://github.com/demorest/guppi_daq
[3] http://psrdada.sourceforge.net/Parkes
[4] http://lwlab.dunlap.utoronto.ca/kotekan/index.html
[5] http://adsabs.harvard.edu/abs/2015ascl.soft07003M

折叠模式和子带模式。

GUPPI 基于 CASPER FPGA 板卡处理 800MHz 带宽、8bit 量化数据，在 10Gb 以太网环境下封装成 UDP 数据包发送，由 GUPPI_daq 负责接收，并将数据保存至磁盘上。传输过程中实现与控制子系统交互，提供人机交互的软件界面，并记录监控参数，保存到输出文件中。

GUPPI_daq 使用多线程模式工作，线程之间通过基于共享内存的环形缓冲区传输数据。在基础模式下，GUPPI_daq 只使用两个处理线程和一个数据缓冲区，由网络处理线程负责接收 FPGA 生成的 UDP 数据包，通过检查自定义的序号来判断数据包是否丢失，将收到的数据保存在环形缓冲区内，丢弃的数据填充为零，同时从状态缓冲区读取头信息，将其复制到数据块头中。写盘处理线程读取已写满的缓冲区并解析数据块头信息，以 PSRFITS 格式保存在磁盘上。

高速率运行时，GUPPI_daq 存在磁盘写入速度的限制问题。通过高速以太网接收数据时，800MB/s 数据接收速率下，出现了少量数据丢包[9]。

1.2 Hashpipe

Hashpipe 是早期 GUPPI_daq 的衍生，最初作为 NRAO VEGAS[10]的高效共享管道引擎，由 David Macmahon 改写，可用于 FX 相关器、波束合成器、脉冲星观测、FRB、SETI，亦可应用于我国 FAST[6]、“天籁”项目[11]。

Hashpipe 设置共享内存段和信号量机制等功能，在运行时基于命令行参数动态构建管道，整体结构如图 1 所示，采用模块化架构设计，蓝色部分表示插件，Hashpipe 插件是一个共享库，定义了应用程序特定的线程模块、共享数据缓冲区等功能模块，并且插件允许用户创建，以便在 Hashpipe 运行时中调用；红色部分表示可执行文件，在 Hashpipe 运行时动态调用已定义插件。

6 https://casper.ssl.berkeley.edu/wiki/images/2/2b/FAST_Hashpipe_Pipeline.pdf

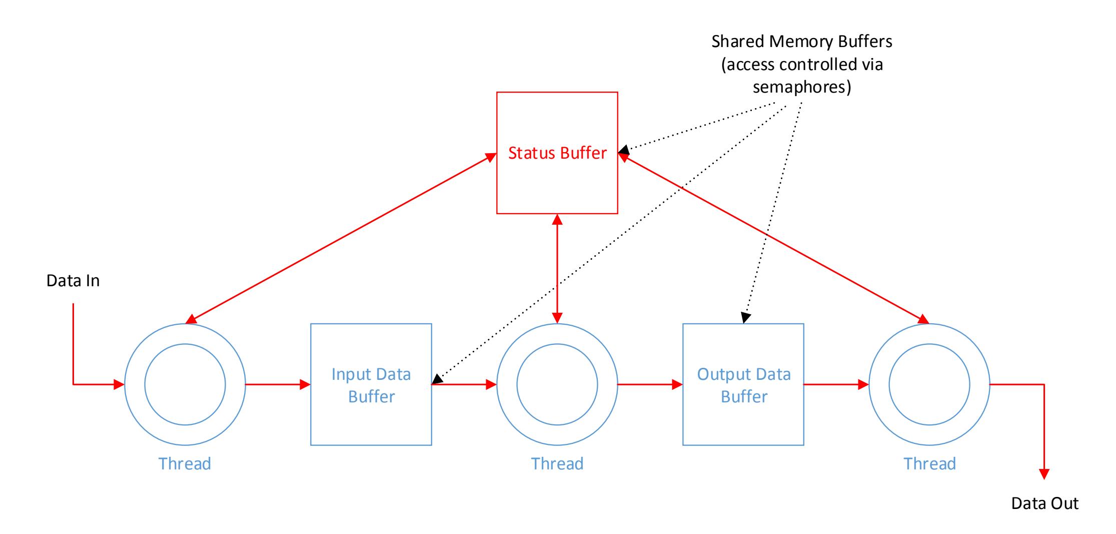


图 1 Hashpipe 整体框架

Figure 1 Hashpipe overall framework

Hashpipe 核心是灵活的环形缓冲区，模拟连续内存块，并实现了数据在多个线程之间的流转和共享，利用 CPU 控制启动和关闭等操作，通过环形缓冲区暂存并传递 UDP 数据包，保证数据快速捕获并按正确顺序分发。

Hashpipe 将任务传输到单独的线程中，每个线程之间共享内存缓冲区，并通过信号量机制控制任务实现线程间互斥。定义数据接收线程、处理线程、输出线程 3 个线程，各个线程实现各自的任务。数据接收线程 net_thread，接收来自计算机万兆网卡的高速网络数据包，根据数据包格式提取文件头和有效数据并解析包头，来自 FPGA 的数据包都有时间戳，如果无序到达，可以重新排列为适当的时间序列，并将数据写入第一个输入数据缓冲区，一旦连续的数据块写满，该块将交由下一个线程处理[12]；处理线程 gpu_thread 从输入数据缓冲区获取数据传输至 GPU 执行复杂计算，然后将结果写入输出数据缓冲区；输出线程 output_thread 从输出数据缓冲区获取数据，将其写入文件存储在磁盘上。通过定义监控线程监听来自 FPGA 的 UDP 数据包，通过转置线程重新格式化数据，使时间样本在内存中正确对齐。

后端由 C/C++编写，头文件中定义多个结构体变量以便调用，定义数据缓冲区 hashpipe_databuf，包含需传输的数据类型、缓冲区空间的大小、缓冲区 ID、数据存储块数、块 ID 等基本信息；hashpipe_thread_desc 结构用于存储描述 Hashpipe 线程的元数据；hashpipe_udp_params 结构体变量，用于保存网络连接参数，包括端口号、IP、数据包的大小等信息。hashpipe_clean_shmem() 函数用来清除状态缓冲区占用；hashpipe_status_unlock_safe()、hashpipe_status_lock_safe()函数用于状态缓冲区的线程安全锁或解锁，确保状态缓冲区处于锁定状态。借助 Hashpipe 已有函数基本满足了数据传输需要，可根据实际需求实现调用。

界面监控采用 David Macmahon 编写的 ruby 程序 rb-Hashpipe，作为 Hashpipe 的可视化

前端可以展示数据包接收、线程状态、缓冲区状态等信息，在更高级别将管道抽象以提供简洁和直观的界面监控数据。Hashpipe 及 rb-Hashpipe 均可部署在服务器平台，环境配置主要步骤如下：

（1）更新 GNU 系列工具到最新版本提供 Hashpipe 编译环境；

（2）安装 Ruby 管理器 rvm；

（3）使用 RubyGems 更新所需库文件；

（4）获取 Hashpipe 版本信息并实现安装；

（5）配置 gnome 终端在 shell 下启动。

实现数据传输状态监控界面如图 2 所示。

图 2 数据传输监控界面

Figure 2 Data transmission monitoring interface

1.3 PSRDADA

PSRDADA 由澳大利亚斯威本科技大学开发维护，支持脉冲星信号分布式采集和数据分析，主要运用于脉冲星基带数据记录和处理，管理从 ADC 信号采样到 GPU 集群数据分析整个过程的数据流。PSRDADA 可实现 APSR、BPSR、CASPSR 基带记录数据分发管理和监控，现阶段主要应用于澳大利亚 Parkes 的 ASPSR、BPSR、CASPSR、HIPSR[13]，欧洲 PuMa-II 等射电望远镜的终端系统[7]。

在基础层次来看，PSRDADA 是一个灵活且易管理的环形缓冲区，其核心功能是在环形缓冲区之间传递数据流。每个缓冲区划分为描述文件信息的文件头存储块，以及多个存储数

[7] https://researchdata.ands.org.au/psrdada-acquisition-distributed-analysis-software/185603

据的子块，接收到数据时按顺序写入子块，当一个子块被填满时，标志位被触发，表示子块中的数据可以读取。利用环形缓冲区实现对数据集进行出队和入队，多个数据集可以利用多线程并行在多个环形缓冲区排队，主要通过共享内存、信号量机制进行线程之间互斥通信。在高级别配置来看，可在集群中各节点启动 PSRDADA 配置，通过可视化界面监控相关流程，管理数据的传输和归档。

PSRDADA 基于模块化设计实现，包括用于执行特定任务的独立进程，整齐分离数据传输、命令、控制指令、数据分析等不同操作；创建环形缓冲区，实现进程之间数据传输；配置文件、脚本等，完成分布式工作流程中各种基础配置；基于 web 的用户接口，提供基于 web 浏览器的监控界面，并可通过 gridbus 发送到远程计算设备实现远程监控。

PSRDADA 基于 C 语言实现了针对 APSR、BPSR、CASPSR、PUMA2 不同设备的相应开发，通过多命令行执行，dada_db 可以创建或删除一个缓冲区，dada_dbmonitor 监控已开辟的缓冲区，dada_dbdisk 命令实现从缓冲区数据写入本地磁盘，dada_header 获取数据文件头信息，dada_dbcopydb、dada_dbdisk、dada_diskdb 实现缓冲区与缓冲区及缓冲区与磁盘间数据拷贝等[8]。

PSRDADA 配置与基本使用步骤如下：

（1）安装或更新 PSRDADA 需要的 GNU 工具 autoconf（2.59 及以上版本），automake（1.9.3 及以上版本），libtool（1.5.8 及以上版本），m4（1.4 及以上版本）；

（2）获取 dada 格式的数据；

（3）创建特定 key 值的环形缓冲区：$dada_db –k key_value；

（4）数据写入环形缓冲区：$dada_diskdb –k key_value –f filename.dada；

（5）从环形缓冲区读出数据：$dada_dbdisk –D . –k key_value –s；

（6）实现数据读写过程中的状态监控：$dada_dbmonitor –k key_value；

（7）删除特定 key 值的环形缓冲区：$dada_db –k key_value –d。

缓冲区创建及数据传输情况如图 3 所示，开辟共享内存键为 100 的缓冲区结构，包括 4 个大小为 524288bytes 的数据存储缓冲区，总容量为 2MB，8 个大小为 4096bytes 的文件头存储缓冲区，总容量为 32KB。dada_dbmonitor 命令打开监控界面，实现对文件头和数据缓冲区读写状态监控。

8 https://github.com/AA-ALERT/psrdada/tree/master/src

```
zhangmeng@ubuntu:~$ dada_db -k 100
Created DADA data block with nbufs=4 bufsz=524288 nread=1
Created DADA header block with nhdrs = 8, hdrsz = 4096 bytes, nread=1
```

```
zhangmeng@ubuntu:~$ dada_dbmonitor -k 100
HEADER BLOCK:
Number of buffers: 8
Buffer size: 4096
Total buffer memory: 32 KB
DATA BLOCK:
Number of readers: 1
Number of buffers: 4
Buffer size: 524288
Total buffer memory: 2 MB

HEADER            DATA
FRE FUL CLR  W  R    FRE FUL CLR    W     R
  7   1   0  1  0      0   4   0    4     0
  7   1   0  1  0      0   4   0    4     0
```

```
7  1  0  1  0    0  4  0    4    0
8  0  1  1  1    4  0  2   65   64
8  0  1  1  1    4  0  3   81   81
8  0  1  1  1    4  0  3   99   99
8  0  1  1  1    4  0  2  113  112
8  0  1  1  1    4  0  3  131  131
8  0  1  1  1    4  0  2  145  144
8  0  1  1  1    2  2  0  171  168
8  0  1  1  1    4  0  3  183  183
8  0  1  1  1    4  0  2  213  212
8  0  1  1  1    4  0  3  247  247
8  0  1  1  1    4  0  2  273  272
8  0  1  1  1    4  0  2  301  300
8  0  1  1  1    4  0  3  323  323
8  0  1  1  1    4  0  3  345  345
8  0  1  1  1    4  0  3  365  365
```

图 **3** PSRDADA 创建数据缓冲区

Figure 3 PSRDADA create data buffers

1.4 Bifrost

Bifrost 专门为射电天文数字信号处理而设计，实现流数据处理高性能管道的快速开发，用于 CPU 与 GPU 之间数据高效传输。现阶段 Bifrost 用于处理 LWA 的数据，实现了在波束合成、相关器上的应用[14]。

Bifrost 利用 Python 实现了高级管道接口，可用于数据传输；利用 C++实现了支持 GPU 的高性能后端，包括一系列与天文数据处理相关的高性能数据结构和算法，为干涉测量、脉冲星消色散、计时及瞬态信号搜寻等应用而设计的库函数，用于 CPU 和 GPU 上数据计算与处理[15]。

其中接口和函数都被称为块，通过灵活的环形缓冲区可以同时实现多个块之间通信，同时执行多线程，使得块可以异步操作。有三种类型的数据块：tasks，从环中读取数据，并实现数据转换，将结果写入到输出环形缓冲区；source，生成数据的块，从文件或以太网数据流的管道外获取数据，将数据写入输出缓冲区；sink，可以直接从输入环形缓冲区读取数据并绘制数据的块，或将其写入文件[9]。

Bifrost 安装步骤简洁，适合快速部署，安装完成后，通过 import bifrost 实现模块调用。例如 bifrost.pipeline 处理管道构建，首先自动创建环形缓冲区，并通过这些环形缓冲器构建有向图，为用户提供高级视图，pipeline.run 语句执行管道，为每个块创建一个线程；通过 bifrost.blocks 加载功能块，调用 FFT、FDMT、量化、转置等实现数据处理；网络相关模块 bifrost.udp，用于捕获 UDP 数据包，可统计并分析包传输信息。如果数据流传输过程中，需

[9] https://ledatelescope.github.io/bifrost/intro.html

要一些中间操作，用户可自定义模块[10]。

以 wav 格式的声音文件为例，Bifrost 执行步骤一般为，读取 wav 文件到环形缓冲区；使用 GPU FFT 实现数据信道化等操作；将 filterbank 文件写入磁盘。

（1）读取.wav 格式的文件；

（2）将原始数据拷贝到 GPU；

（3）将时间轴分割成小块，对新生成的小块执行 FFT；

（4）取这些 FFT 模的平方；

（5）将数据转置为 sigproc 兼容的格式，进行标准化；

（6）将数据拷贝到 CPU；

（7）将数据转换为整数型，并将数据以 filterbank 存储。

1.5 Kotekan

Kotekan[16]是 Andre Recnik 使用 C/C++组合开发的软件框架，主要应用于射电天文望远镜数据传输，最早为 CHIME 后端应用开发，旨在实现更高的效率和吞吐量[11]。

Kotekan 管道是模块化的，由一系列对数据执行不同操作的单元组成，实现了实时管理 X 引擎上数据流，包括接收来自 F 引擎的 UDP 数据包，GPU 结点内数据流实时处理及结果存储。核心结构是 FIFO 环形缓冲区，通过系统内存中环形缓冲区，实现了 CPU 与 AMD GPU 之间通信，及传输到 GPU 进行处理之前数据检查和暂存[17]。

Kotekan 创建网络线程、GPU 线程、GPU 回调线程、GPU 后处理线程和输出线程，每个线程都与一个或多个环形缓冲区对象的接口相关联，接收到的数据均通过环形缓冲区暂存，实现边读边写，这些线程和缓冲区形成了如图 4 所示的数据路径，管道可以多线程并行的产生多个结果数据集[18]。

[10] https://github.com/ledatelescope/bifrost
[11] http://lwlab.dunlap.utoronto.ca/kotekan/overview.html

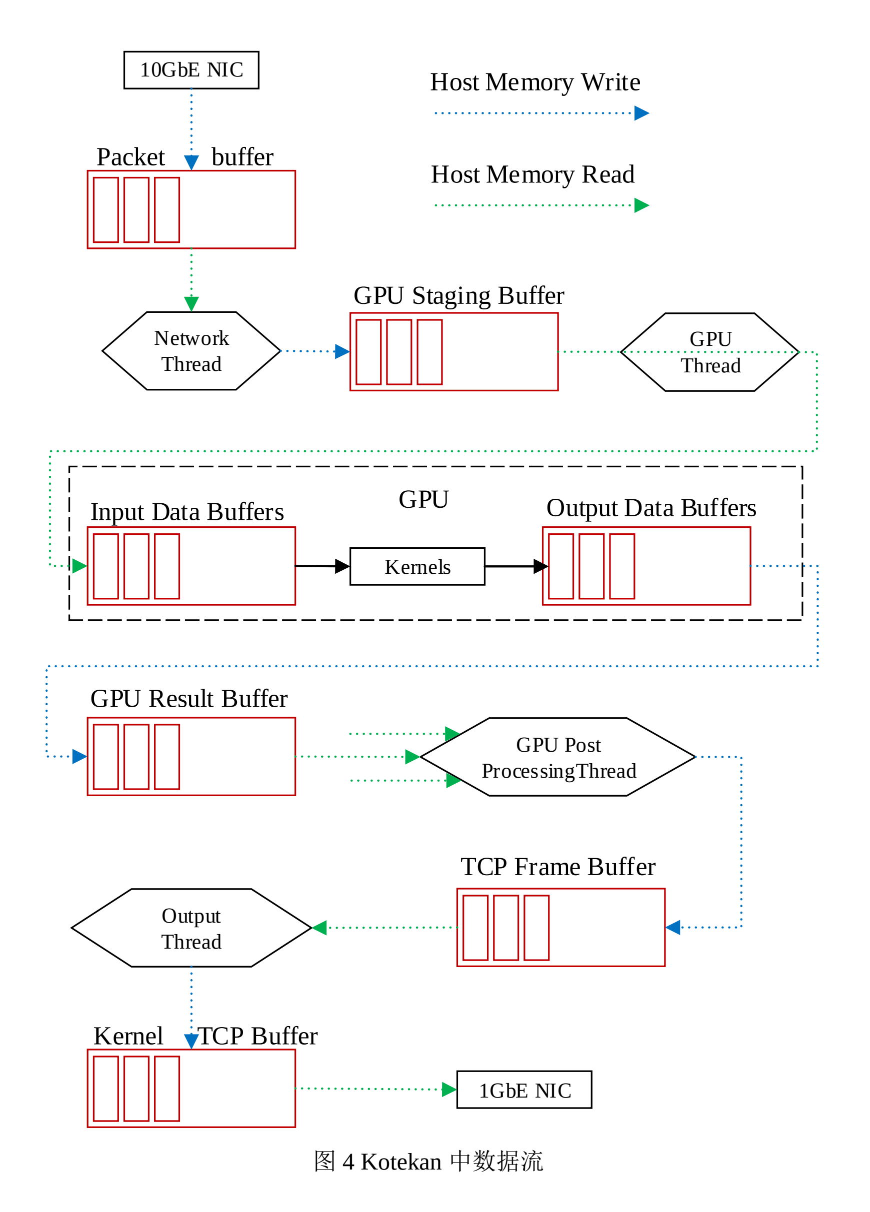


图 4 Kotekan 中数据流

Figure 4 Data flow in Kotekan

缓冲数据由 Kotekan::stage 处理模块写入和读取，从网络或外部设备获取数据，通过数据处理算法实现数据操作，最后将处理后的数据通过网络传输或存储于本地。

1.6 Pelican

Pelican[19]是由牛津电子中心开发的实时数据流处理框架，具备高效、模块化、轻量级、可校准的特点，主要应用于 LOFAR 和 BEST-II，通过 GPU 服务器对数据进行并行处理。Pelican 提供了高度抽象的 API，适合用于静态、准实时处理。

Pelican 用 C++实现了可重用的模块化组件，分离了数据的获取和处理，具有灵活性和通用性，具备读取 UDP 数据的机制，将数据流传递给各计算服务器，完成数据缓冲、处理和最后文件写入，可对望远镜接收数据进行实时处理，也可用于任何需要处理传入数据流的

应用程序[20]。

Pelican 可用于脉冲星搜寻的数据预处理，SKA 早期计划使用 Pelican 处理非成图的基于 GPU 实时射电脉冲检测[21]，流程如图 5 所示，首先通过软件服务器读取 UDP 数据，去除窄带频谱干扰的峰值，实现 GPU 中进一步窄带划分，完成复杂数据到功率数据的转换，将数据传输至 GPU 处理模块执行分散搜索算法[22]。

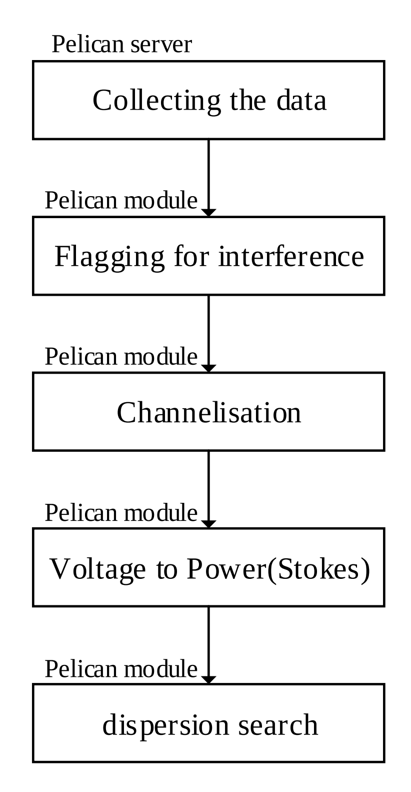


图 5 SKA 计划利用 Pelican 实现数据处理流程

Figure 5 SKA plans to use Pelican for data processing

短时间内成像导致非常大的相关器输出速率，为了实现数据传输速率并产生更新的校准系数，需对输出数据流进行实时处理。Pelican 可用于实现射电望远镜数据转换成科学图像之前的数据校准和成像方程求解。Pelican 实现科学成像的框架如图 6 所示，接收来自 ROACH2 的子带数据，通过实时更新相关器的相位和振幅系数，提供初始校准，然后基于 GPU 的 2D-FFT 生成脏图像，洁化后执行基于特定步长的图像差分比较，阈值检测器用来发现瞬态事件，最后堆叠模块用来形成一个高动态范围的图像[12]。

[12] http://www-astro.physics.ox.ac.uk/~FosterG/ppp/lofar_poster_ursi_03.pptx

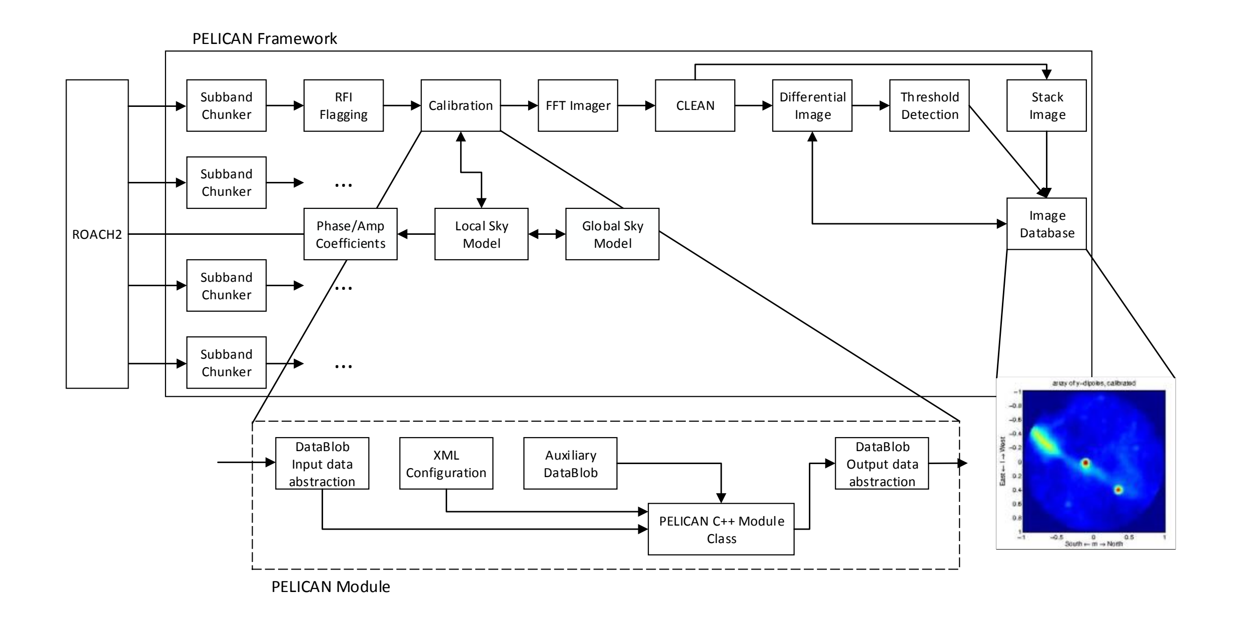


图 6 Pelican 实现成像框架

Figure 6 Pelican realized imaging framework

1.7 Cobalt

Cobalt 是基于 GPU 设计与开发的 LOFAR 望远镜软件相关器和波束形成器[23]，软件部分实现了数据管道，用于管理通过网络和 CPU、GPU 的数据流，并完成数据存储。

图 7 为数据流从外部系统到 Cobalt 内部数据流示意图，接收数据存入输入缓冲区，完成子带处理后进行存储，模块间以 all-to-all 模式进行通信。

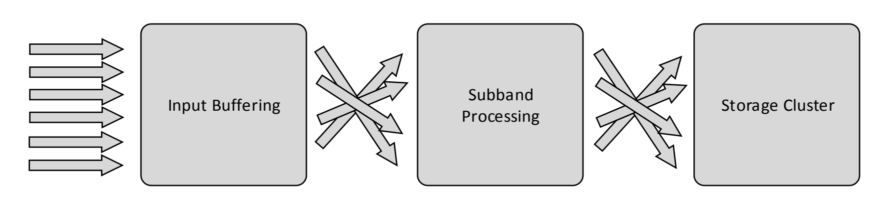


图 7 Cobalt 数据流示意图

Figure 7 Cobalt data flow

Cobalt 的软件由几个交互子系统组成，管理来自网络的数据流，实现高速率、可持续数据传输，并持久性存储数据；进行控制、监控、记录日志和元数据。图 8 显示了 Cobalt 的组件、依赖关系和数据流，Cobalt 处理大量独立的数据流，每个 AntennaFieldInput 接收 10GbE 端口的 UDP 数据包，并将有效数据转发到 TransposeSender，通过环形缓冲区移动整数倍执行延迟补偿。这些数据流在没有相互依赖的情况下结合 MPI 实现并行处理，GPU 上的信号处理管道 GPUpipeline 包含 TransposeReceiver 组件，传输子带数据，与 MPI_Isend 和 MPI_Irecv

非阻塞发送和接收相结合传输相关或波束数据。输出组件接收 GPU 子带数据，最终使用 casacore 以 MeasurementSet 或 HDF5 格式存储[24]。

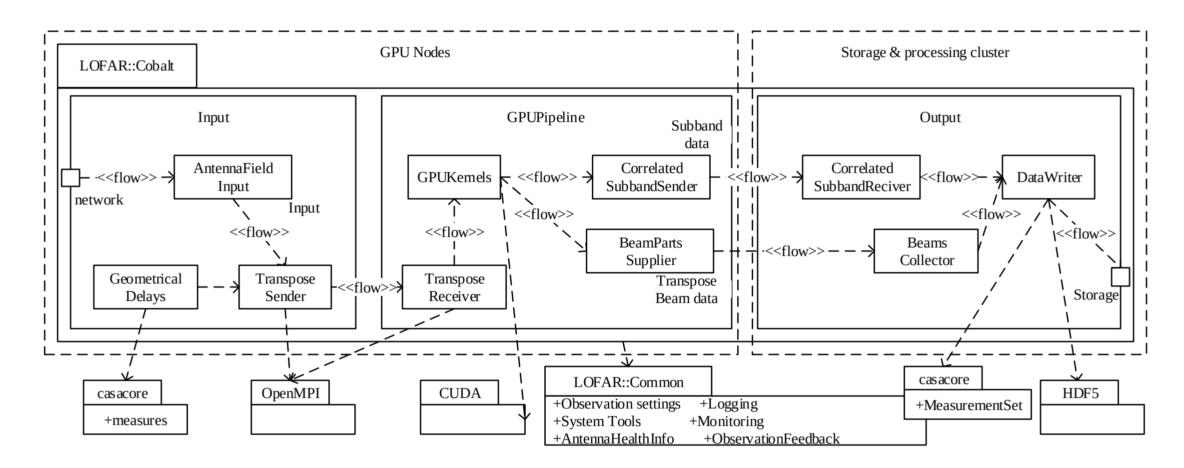


图 8 Cobalt 组件依赖关系和数据流

Figure 8 Cobalt component dependencies and data flow

Cobalt 可以处理离线任务，包括自动标记、校准、平均、脉冲星数据相干消色散和动态光谱产生以及脉冲星数据在线折叠和搜索。Cobalt 具备可扩展性，如子阵列并行观测、中断响应以及其他的观测模式。在网络带宽和计算能力方面，提供了显著的额外容量，效率有显著的提高，例如低波段天线和高波段天线的并行观测。一般来说相关器只能处理单个项目的数据，Cobalt 因拥有巨大的计算和吞吐能力可以同时处理多个项目数据[13]。

现已有更新版本 Cobalt 2.0，具备更好的灵活性和吞吐量，接收数据量速率超过 1TB/s，更大限度利用 GPU 的优势，实现高性能计算和数据实时处理。

2 面临的问题

现阶段数据流管道软件面临的重点与难点：

（1）现有射电天文数据流管道软件均针对特定观测设备设计，为满足特定需求开发，并不适用于所有类型的观测设备及观测模式。针对具体观测设备，应根据各观测设备的接收系统需求来实现相应的软件功能，部分软件系统虽然可移植到不同的观测设备终端系统，使用时并不需要其全部功能，需要深入研究其底层代码，完成二次开发及调试，开发成本高、难度大。

[13] https://www.astron.nl/what-we-look-forward-to-in-lofar-2-0-a-brain-transplant-for-lofar/

（2）已有大型射电望远镜终端系统，多采用基于 Xilinx FPGA 的 ROACH 系列、SNAP 系列硬件板卡，近几年推出了集成度更高的 RFSoC，可提供高速以太网数据传输接口，对服务器端数据流实时接收能力提出了更高的要求。部署在服务器端的管道需要满足基于 RFSoC 平台更高带宽的数据接收需求，提升数据传输效率。

（3）数据实时高效传输与处理能力是管道软件性能提升方向，数据处理平台多采用 GPU 集群实现大规模数据处理，然而数据分发与收集并没有得到有效的处理，高效实现数据流在多 GPU 上的分发与汇总，将大大提高传输与计算效率。

（4）为满足多功能数字终端系统的数据处理要求，应形成一套配置灵活的系统软件，构建的管道软件应实现数据处理与分析所需算法，针对不同观测模式提供可调用的类库，满足多观测课题的需求。

3 大口径射电望远镜数据流管道软件设计思路

应用于大口径射电望远镜数据流传输与处理的管道软件，作为多功能终端系统软件中重要的组成部分，将完成数据流获取、传输、管理与监控。高效捕获并暂存来自 FPGA 的网络数据包，实现数据在 CPU 与 GPU 之间高速流转及相关处理，提供 GPU 内部数据处理算法调用，实现原始数据格式到标准文件格式的转换及数据类型的匹配，最终数据高效写入磁盘存储。

3.1 基于零拷贝和环形缓冲区的数据传输

QTT 多功能数字终端采用 FPGA+GPU 集群架构模式，如图 9 所示[25]。RFSoC 实现数据采样和预处理，单块 RFSoC 芯片可实现 8 通道，每通道最大采样速率 4.096GSPS，12bit 采样，采样后每通道的数据量约为 48Gbps，经过 RFSoC 预处理将采样数据划分成多个子带，利用 RFSoC 提供 100Gb 的数据传输接口，通过高速以太网传输至 GPU 集群实现数据相关联进行数据预处理。

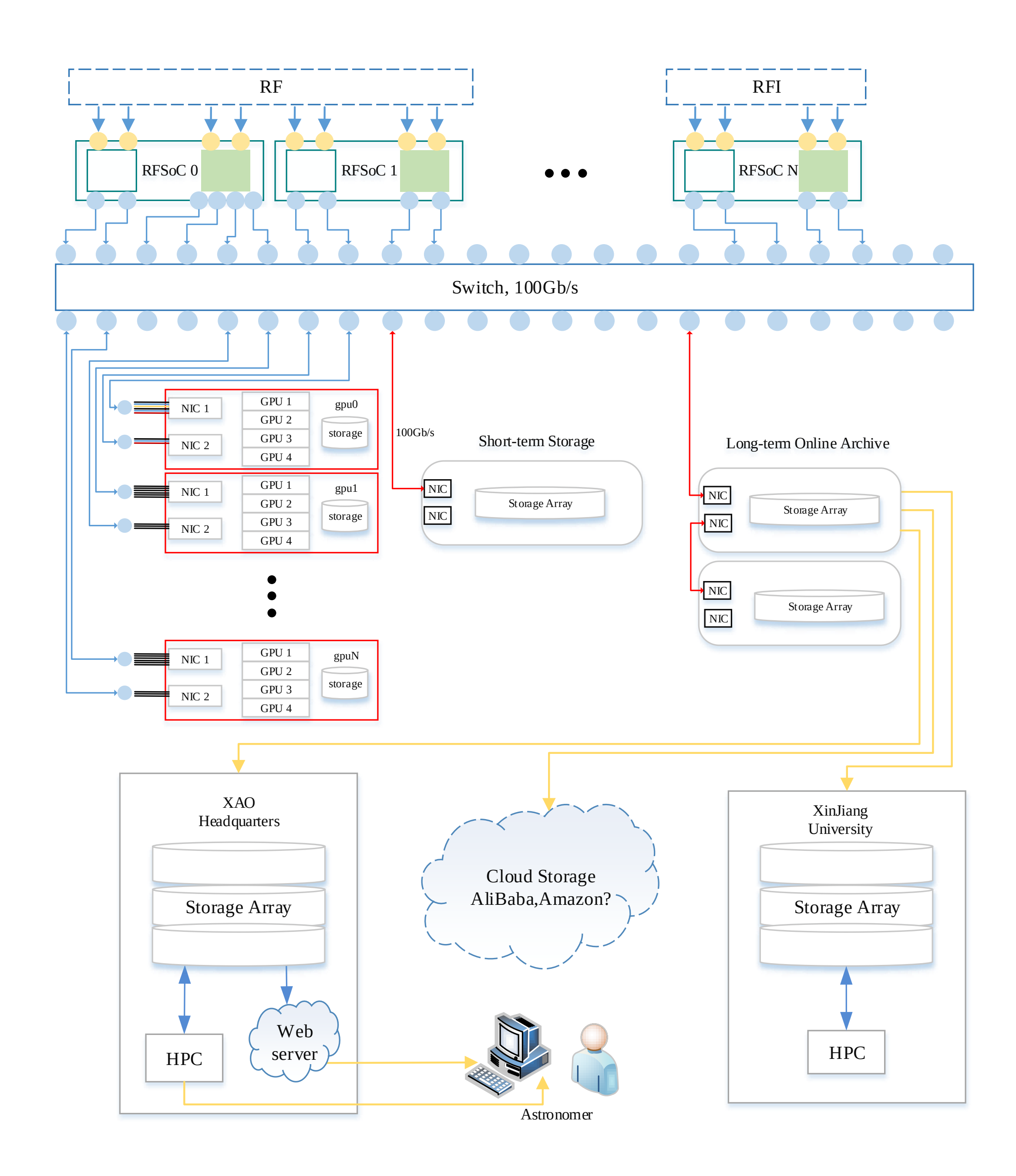


图 9　QTT 终端系统框架

Figure 9 The block diagram of QTT backend system

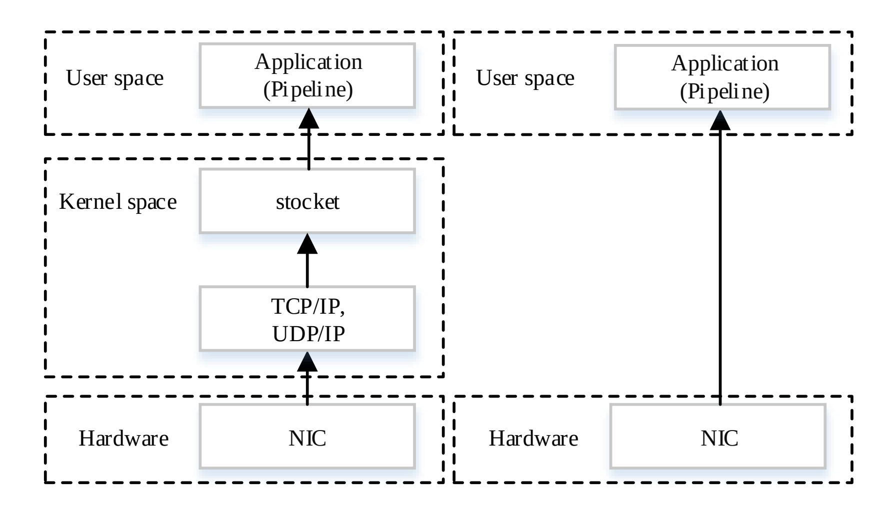


图 10 标准数据传输与零拷贝数据传输

Figure 10 Standard data transfer and zero-copy data transfer

网络数据传输使用 UDP 协议，网卡缓冲区中数据通过 Linux 内核协议栈传输至用户应用程序，数据包在内核态与用户态之间拷贝，系统消耗大且存在传输瓶颈，为了有效降低系统资源占用，满足 GPU 服务器端传输的低丢包率和低延迟，GPU 服务器端可基于零拷贝技术实现高 I/O 环境下的高性能数据获取，对比如图 10 所示。管道软件实现基于零拷贝的数据传输模块，创建并行网络输入线程在用户态以轮询方式读取网卡缓冲区中数据包，实时写入动态创建的环形缓冲区，避免标准内核协议栈的处理，减少网卡到管道软件过程中的数据拷贝次数，以提高数据访问效率。

3.2 基于 GPU 集群的数据实时处理

基于混合架构的 QTT 多功能终端系统计划实现脉冲星、暂现源、分子谱线、总功率、VLBI、基带数据等多种观测模式，GPU 集群内部数据流如图 10 所示，FPGA 预处理后的数据传输至 GPU 集群实现数据处理，数据流经解码、RFI 消减后，针对不同观测模式完成相应处理，并最终以 VDIF、PSRDada 或 PSRFITS 等格式存储。

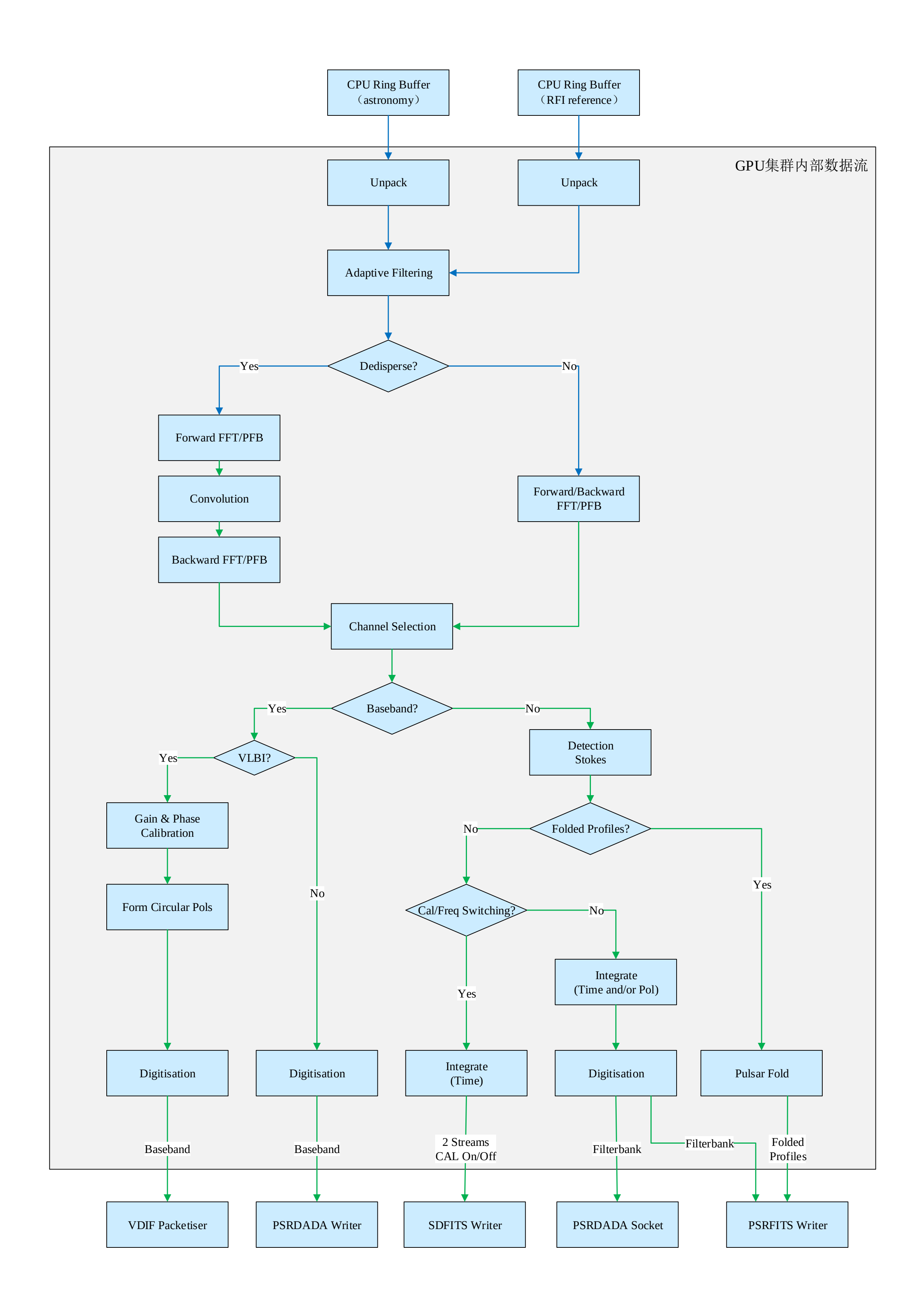


图 10　QTT 多功能数字终端 GPU 集群内部数据流

Figure10 QTT multifunctional digital backend GPU cluster internal data flow

管道软件应提供 GPU 内部数据处理模块，并应充分整合天文学算法，实现解码、RFI 识别、校准、折叠、标准格式输出等常用功能，实现到标准文件格式的转换及数据类型的匹配。基于 Web 提供友好的访问界面，以模块化的方式提供更好的灵活性并支持扩展，提供用户自定义模块功能，允许用户根据实际需求调用模块或编写自定义模块。

针对 QTT 多功能数字终端数据流，管道软件数据传输过程的顶层结构如图 11 所示。网络线程捕获来自 RFSoC 的数据并存放至输入数据环形缓冲区，管道软件运行在多台 GPU 服务器上，每台 GPU 服务器接收一个频率通道子带，为实现 GPU 集群上多线程并行和分布式执行，高效及时运行和数据块转发，集群节点中结合 MPI 实现数据的分发和获取，采用 MPI 提供的非阻塞消息接收与发送调用接口 MPI_Isend、MPI_Irecv，实现计算和通信重叠进行，提高数据处理效率。GPU 集群处理后数据暂存于输出数据缓冲区，并通过并行输出线程传输至存储系统。

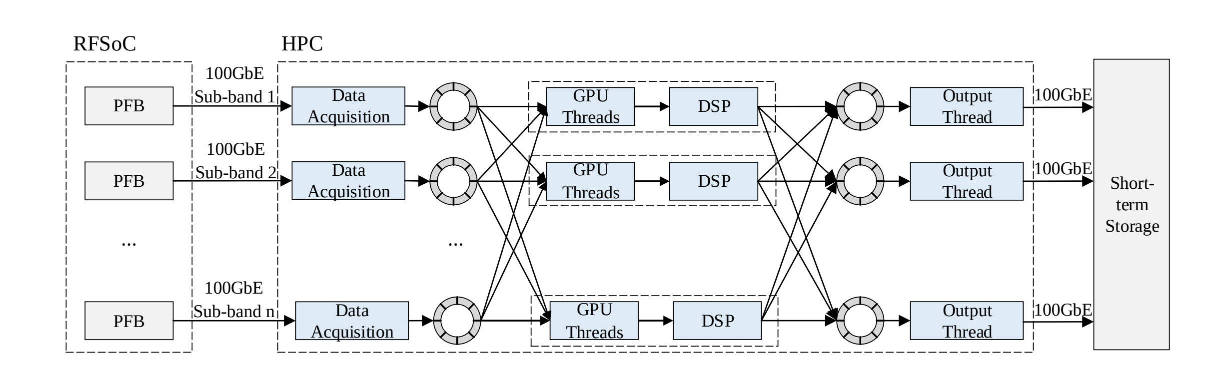


图 11 数据传输流程图

Figure 11 Data transmission flowchart

3.3 基于 BeeGFS 的数据存储

随着数据量的迅速增加，HPC 系统需要更复杂、更高效的方法管理与存储大量数据。通过部署分布式文件系统将数据分布到众多存储设备上，对数据进行分布式存储和管理，实现分布式读写。借助并行分布式文件系统 BeeGFS[14]构建更高效的数据存储后端，提高 I/O 性能，实现高性能和可扩展性。

[14] https://www.beegfs.io/c/

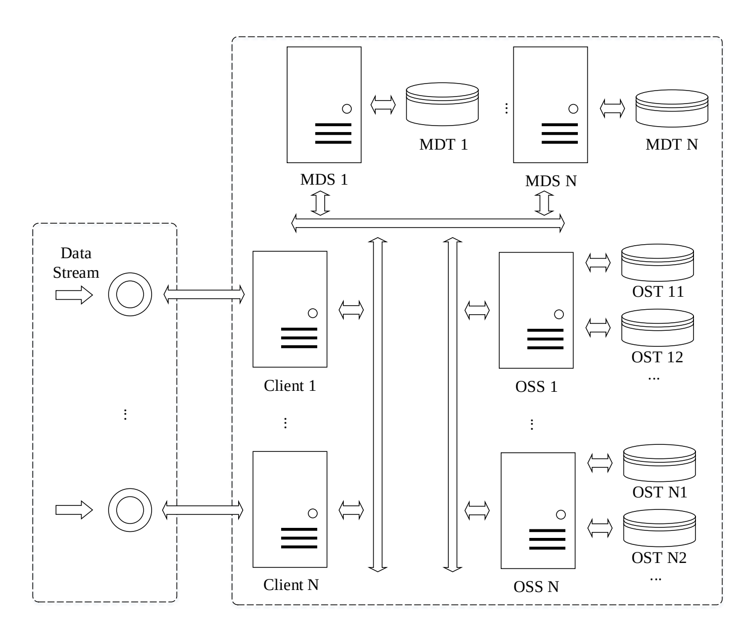


图 12 基于 BeeGFS 的数据存储系统

Figure 12 Data storage system based on BeeGFS

基于 BeeGFS 的数据存储系统整体框架如图 12 所示，GPU 集群处理后的数据，进行整合后封装成特定的格式，通过构建并行输出管理线程同时传输一个或者多个数据流，相同格式的数据写入同一环形缓冲区，传输至文件存储系统，通过 BeeGFS 提供的 Client 读写数据，利用多个独立 I/O 实现高并发读写。基于 BeeGFS 分布式文件系统实现元数据分离，分别利用 MDS、OSS 的本地文件系统承载元数据和数据的存储，其中文件头中包含的日期时间、观测环境参数等观测信息，作为元数据信息的主要组成部分，存储于元数据服务器。MDS、OSS 以及 Client 的扩展性使得基于 BeeGFS 数据存储系统能够满足大规模数据存储扩展需求。

4 结语

射电天文望远镜数字终端系统越来越依赖实时处理来克服数据存储和分析的瓶颈，天文领域已经开发并研究了数据流处理管道，用以提高数据传输和处理的效率，简化天文学家数据处理代码开发难度。文章综述了现有射电天文数据传输管道软件，介绍了射电天文数据传输管道软件的结构、功能，并在 GPU 服务器上进行了测试。在总结现有管道软件的基础上，给出了未来大口径射电望远镜多功能终端系统应根据实际观测需求设计数据传输管道的一般思路，未来多功能数字终端系统基于 FPGA+GPU 实现，应该向易升级、改造、灵活、可扩展的方向发展。

致谢：感谢国家自然科学基金(11873082,11803080,12003062)；中国科学院青年创新促进会；国家重点研发计划(2018YFA0404704)；天文财政专项；国家天文科学数据中心，中科院科学数据中心体系的资助。本论文得到中国虚拟天文台、国家天文科学数据中心、中科院科学数据中心体系提供的数据资源和技术支持。

5 参考文献

[1] 张萌, 张海龙, 王杰, 李健, 托乎提努尔. 基于 ROACH2 的数字终端实验平台搭建[J]. 天文研究与技术, 2020, 17(02):244-251.

Zhang Meng, Zhang Hailong, Wang Jie, Li Jian, Tohtonur. The Construction of Digital Backend Experiment Platform Based on ROACH2[J]. Astronomical Research & Technolog, 2020,17(02):244-251.

[2] Farley B, Mcgrath J, Erdmann C. An All-Programmable 16-nm RFSoC for Digital-RF Communications[J]. IEEE Micro, 2018, 38(2):61-71.

[3] Paine D, Lee C P. Producing data, producing software: Developing a radio astronomy research infrastructure[C]//2014 IEEE 10th I nternational Conference on e-Science. IEEE, 2014, 1: 231-238.

[4] Wang N. Xinjiang Qitai 110 m radio telescope[J]. entia Sinica, 2014, 44(8):783.

[5] 张海龙，王杰，王万琼，聂俊，冶鑫晨，朱艳，托乎提努尔. 新疆天文台数据中心建设与应用[J]. 天文研究与技术, 2017, 14(01):94-102.

Zhang Hailong,Wang Jie,Wang Wanqiong,Nie Jun,Ye Xinchen,Zhu Yan, Tohtonur. Construction and Application of the Data Center in Xinjiang Astronomical Observatory[J]. Astronomical Research & Technolog, 2017,14(01):94-102.

[6] Srikanth S, Norrod R, King L, et al. An overview of the Green Bank Telescope[C]// Antennas & Propagation Society International Symposium. IEEE, 2002.

[7] 梅丽, 苏彦, 周建锋. 极低频射电天文观测现状与未来发展[J]. 天文研究与技术, 2018, 15(02):127-139.

Mei Li, Su Yan, Zhou Jianfeng. The History and Development of The Low-Frequency Radio Observation[J]. Astronomical Research & Technolog, 2018,15(02):127-139.

[8] Bandura K, Addison G E, Amiri M, et al. Canadian hydrogen intensity mapping experiment (CHIME) pathfinder[J]. Proceedings of SPIE - The International Society for Optical Engineering, 2014, 9145.

[9] Duplain R, Ransom S, Demorest P, et al. Launching GUPPI: the Green Bank Ultimate Pulsar Processing Instrument[J]. 2008, 7019:70191D.

[10] Prestage R M, Bloss M, Brandt J, et al. The versatile gbt astronomical spectrometer (vegas): Current status and future plans[C]//2015 USNC-URSI Radio Science Meeting (Joint with AP-S Symposium). IEEE, 2015: 294-294.

[11] Chen-Hui Niu,Qun-Xiong Wang,David MacMahon,Feng-Quan Wu,Xue-Lei Chen,Ji-Xia Li,Hai-Jun Tian,Guillaume Shippee,Dan Werthimer,Xiao-Ping Zheng.The design and implementation of a ROACH2+GPU based correlator on the Tianlai dish array[J].Research in Astronomy and Astrophysics,2019,19(07):135-144.

[12] Macmahon D H E, Price D C, Lebofsky M, et al. The Breakthrough Listen Search for Intelligent Life: A Wideband Data Recorder System for the Robert C. Byrd Green Bank Telescope[J]. Publications of the Astronomical Society of the Pacific, 2017, 130(986): 044502.

[13] Price D C, Staveley-Smith L, Bailes M, et al. HIPSR: A Digital Signal Processor for the Parkes 21-cm Multibeam Receiver[J]. Journal of Astronomical Instrumentation, 2016, 5.

[14] Cranmer M D, Barsdell B R, Price D C, et al. Bifrost: a Python/C++ Framework for High-Throughput Stream Processing in Astronomy[J]. Journal of Astronomical Instrumentation, 2017.

[15] Kaszyk K, Wagstaff H, Spink T, et al. Full-system simulation of mobile cpu/gpu platforms[C]//2019 IEEE International

Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2019: 68-78.

[16] Recnik A, Bandura K, Denman N, et al. An Efficient Real-time Data Pipeline for the CHIME Pathfinder Radio Telescope X-Engine[C]// 2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE, 2015.

[17] Pinsonneault Marotte T. Towards precision measurements of the Hubble constant with the Canadian Hydrogen Intensity Mapping Experiment[D]. University of British Columbia, 2018.

[18] Denman N, Amiri M, Bandura K, et al. A GPU-based Correlator X-engine Implemented on the CHIME Pathfinder[J]. 2015.

[19] Mort B, Dulwich F, Williams C, et al. Pelican: Pipeline for Extensible, Lightweight Imaging and CAlibratioN[J]. Astrophysics Source Code Library, 2015.

[20] Karastergiou A, Chennamangalam J, Armour W, et al. Limits on fast radio bursts at 145 MHz with ARTEMIS, a real-time software backend[J]. Monthly Notices of the Royal Astronomical Society, 2015, 452(2): 1254-1262.

[21] Trefethen A E. Extreme Computing: Challenges, Constraints and Opportunities[J]. 2010.

[22] Karastergiou A. SKA NON IMAGING PROCESSING CONCEPT DESCRIPTION: GPU PROCESSING FOR REAL-TIME ISO LATED RADIO PULSE DETE-CTION[J]. 2011.

[23] Broekema P C, Mol J J D, Nijboer R, et al. Cobalt: A GPU-based correlator and beamformer for LOFAR[J]. Astronomy and computing, 2018, 23: 180-192.

[24] Prasad P, Huizinga F, Kooistra E, et al. The AARTFAAC all-sky monitor: System design and implementation[J]. Journal of Astronomical Instrumentation, 2016, 5(04): 1641008.

[25] 张海龙, 张萌, 聂俊, 等. 脉冲星数字终端技术综述[J]. 中国科学:物理学, 力学, 天文学, 2019(9):15-28.
Zhang Hailong, Zhang Meng, Nie Jun, et al. Pulsar digital backend technologies review[J]. SCIENTIA SINICA Physica, Mechanica & Astronomica, 2019(9): 15-28.

Efficient real-time data transmission and processing technologies applied to radio astronomy

Zhang Meng[1,2], Zhang Hailong[1,2,3,4*], Wang Jie[1,2,4], Li Jian[1], Ye Xinchen[1,4], Wang Wanqiong[1], Li Jia[1], Wang Boqun[1,2], Zhang Yazhou[1,2]

(1. Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011, China;

2. University of Chinese Academy of Sciences, Beijing 100049, China;

3. Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing 210008, China;

4. National Astronomical Data Center, Beijing 100101, China)

Abstract: Aiming at problems of massive signals real-time transmission and processing in ultra-wideband and multi-beam receiving systems, tested and analyzed the related software of the mainstream backend system based on FPGA+GPU, the ultra-wide band receiving equipment requires the backend system software to be capable of wider bandwidth and higher time resolution and higher frequency resolution conditions, realized real-time data stream transmission and processing. Combined with the future development direction of large aperture radio observation equipment, it was proposed to use high-speed parallel ring buffers to achieve data stream caching, GPU clusters to achieve data stream real-time processing, and distributed parallel data storage based on BeeGFS, modular construction of data stream transmission pipelines software design ideas.

Key words: Data transmission and processing; Radio astronomy; Real-time; FPGA+GPU